PENGECEKAN PLAGIARISME PADA CODE DALAM BAHASA C++

Liliana[1], Gregorius Satia Budhi[2], Anthony Wibisono[3], Ricky Tanojo[4]
[1,2,3,4]Program Studi Teknik Informatika, Fakultas Teknologi Indutri, Universitas Kristen Petra Surabaya
Email: lilian@petra.ac.id, greg@petra.ac.id

Abstrak: Program Studi Teknik Informatika UK Petra mempunyai studio pemrograman yang digunakan untuk sarana berlatih pemrograman bagi mahasiswa, khususnya mahasiswa yang mengambil matakuliah algoritma dan pemrograman. Sistem yang saat ini diterapkan adalah memberikan soal-soal latihan sebanyak mungkin untuk dikerjakan sesuai kemampuan mahasiswa. Penilaian dilakukan dengan menghitung poin nilai dari soal-soal yang dikerjakan dengan benar. Permasalahan yang terjadi pada saat pengerjaan soal di studio adalah adanya plagiarisme di antara hasil pekerjaan mahasiswa. Plagiarisme ini membutuhkan banyak waktu untuk mengeceknya. Dalam satu semester seorang mahasiswa AP dapat mengerjakan sekitar 100 soal. Jika total mahasiswa yang mengambil MK AP setidaknya 100 orang, maka butuh kecermatan dan waktu yang lama untuk mengoreksi semua jawaban tersebut. Berangkat dari kesulitan itu, maka dibuatlah sebuah aplikasi untuk mengecek plagiarisme antara dua buah jawaban yang berbeda dari soal yang sama. Tingkat kemiripan dihitung dengan menggunakan gabungan metode jaccard similarity coefficient dan cosine similarity coefficient. Sistem sudah mampu melakukan pengecekan kesamaan pada file-file source code hasil pekerjaan mahasiswa dengan tingkat kompleksitas yang sedang.

Kata kunci: source code, plagiarism, similarity, jaccard similarity coefficient, cosine similarity coefficient

Abstract: *Informatics department of Petra Christian University has used to practice programming for students at programming studio, especially students who take algorithms and programming course. The system is currently implemented is to provide practice programming skill according to their ability. Assessment is done by calculating the points of all questions are done correctly. Problem which occurs at about the work in programming studio is plagiarism among students' work. Unfortunately, plagiarism checking consumes a lot of time. In one semester, each student can do about 100 questions. If the total of students taking programming course, at least 100 people, we need many people to do this work. Departing from this problem, we design and develop an application to check for plagiarism between two different answer files from the same question. The degree of similarity is calculated using combined method of Jaccard similarity coefficient and cosine similarity coefficient. This system is able to check similarities in source code files are the work of students with moderate levels of complexity.*

Keywords*:* source code, plagiarism, similarity, jaccard similarity coefficient, cosine similarity coefficient.

PENDAHULUAN

Algoritma dan Pemrograman merupakan mata kuliah yang wajib diambil oleh setiap mahasiswa semester 1 teknik informatika. Mata kuliah ini merupakan mata kuliah dasar dari banyak mata kuliah lainnya. Untuk bisa menguasai mata kuliah ini, bukan hanya mengerti teori namun harus latihan setiap hari. Dari metode pengajaran yang diterapkan, mahasiswa dikondisikan untuk berlatih mandiri. Sarana yang digunakan adalah mengerjakan kumpulan soal di studio pemrograman.

Permasalahan yang sering timbul adalah terjadinya plagiarisme dalam pengerjaan soal-soal tersebut. Jika ada 150 orang mahasiswa yang mengambil mata kuliah ini, dan jika setiap kali diberikan rata-rata 5 soal, maka harus dilakukan pengecekan untuk 750 jawaban. Untuk melakukan hal ini, dibutuhkan sumber daya dan waktu yang banyak. Untuk mengantisipasi masalah tersebut, didesain sebuah aplikasi untuk melakukan pengecekan secara otomatis. Aplikasi ini akan melakukan pengecekan code mana saja yang merupakan plagiat dari code lainnya. Aplikasi dibangun dengan berbasis web dengan GNU Compiler [1, 2]

SOURCE CODE PLAGIARISM DETECTION

Code Plagiarism adalah istilah yang digunakan untuk mendeskripsikan adanya kemiripan isi file *source code* antar 2 file yang berbeda [3, 4, 5]. Kemiripan dapat terjadi secara keseluruhan file maupun sebagian, dengan sedikit perubahan pada bagian yang tidak signifikan. Terdapat 3 tipe kemiripan *textual* yang dapat terjadi pada dua buah *source code* [4, 5]:

a. *Type I*
   Fragmen yang di-*copy* dari file original secara sama persis, kecuali adanya perbedaan– perbedaan seperti *white spaces* (spasi atau enter yang tidak

mempunyai arti), *comment*, dan modifikasi pada jumlah baris.Tipe ini disebut juga sebagai *exact clone*.

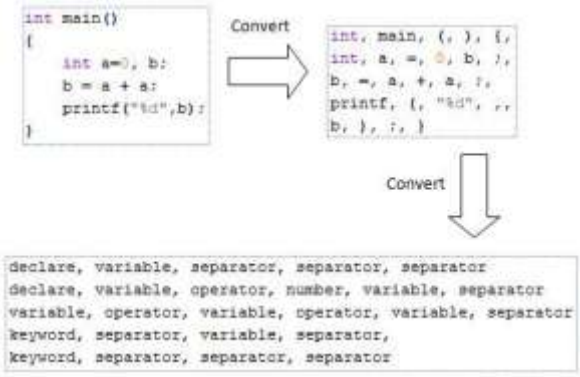
b. *Type II*
Tipe ini sama seperti *Type I*, perbedaannya adalah pada *Type II*, terdapat modifikasi pada variabel atau fungsi. Modifikasi dapat berupa penggantian nama atau perubahan tipe data.

c. *Type III*
Type III merupakan kombinasi dari *Type I* dan *Type II*. Plagiarisme file dilakukan dengan cara menambahkan *statement* yang tidak penting.

Terdapat 4 fase yang perlu dilakukan untuk mendapatkan nilai kesamaan dari 2 file, yaitu [3, 4, 5]:

a. Menghilangkan *white spaces*, *comment* dan *include statement*
White spaces berupa spasi berkepanjangan yang tidak diperlukan atau adanya baris baru yang tidak berisi *statement*. Tanpa *white spaces* maka *source code* akan hanya berisi code yang mempunyai arti untuk mempercepat dan meningkatkan akurasi pada proses *tokenization*.
Comment berisi baris code yang tidak dieksekusi, berupa satu baris atau blok dari beberapa baris. *Comment* satu baris ditandai dengan adanya karakter "//", sedangkan *comment* beberapa baris diawali dengan karakter "/*" dan diakhiri dengan "*/". *Comment* tidak akan dimasukkan dalam proses *tokenization*.
Include statement terletak pada bagian atas *source code* dan diawali dengan karakter "#". Sebanyak apapun *include statement* yang ada dalam suatu source code tidak akan mempengaruhi hasil program sehingga *include statement* tersebut tidak perlu dimasukkan dalam proses *tokenization*.

b. *Tokenization*
Dalam fase ini, *source code* yang sudah diproses di fase sebelumnya akan diubah menjadi serangkaian *token*. *Token* merupakan hasil pengkategorian dari setiap kata atau karakter yang terdapat pada *source code*. Semakin banyak kategori yang ada, akan semakin akurat juga hasil dari *plagiarism detection*. Hasil akhir dari proses *tokenization* ini adalah serangkaian angka di mana masing – masing angka tersebut melambangkan isi dari *source code*. Serangkaian angka tersebut yang nantinya akan dipakai untuk fase berikutnya. Contoh proses *tokenization* dapat dilihat pada Gambar 1.
Rangkaian *token* tersebut akhirnya akan diubah menjadi rangkaian angka dengan menggunakan Tabel 1.



Gambar 1. Proses tokenization

Tabel 1. Tabel token yang digunakan dalam aplikasi

Token	ID
Keywords	1
Separators	2
Operators	3
Declare	4
Variables	5
Numbers	6

Hasil akhir yang didapat adalah seperti berikut ini

45222453652535352125252222

c. *N – gram*
N – gram adalah rangkaian *substring* yang berukuran *N* yang didapatkan melalui perpotongan dari *token*. Sebagai contoh apabila terdapat deretan angka 123456789, maka 5–*gram* dari rangkaian angka tersebut adalah {12345}, {23456}, {34567}, {45678}, {56789}. Semakin besar *N* maka nilai akhir kemiripan antar 2 file akan semakin tidak akurat, akan tetapi semakin kecil *N* maka proses akan semakin lama. Secara umum, besar *N* yang digunakan adalah 4.
Keuntungan dalam menjalankan proses *N – gram* ini adalah perubahan pada suatu *token* hanya akan berakibat pada sedikit *token – token* yang terdapat di sebelahnya. Hal ini juga mengurangi masalah yang terjadi apabila terdapat *statement* tambahan yang digunakan hanya untuk menyamarkan file hasil plagiat.

d. *Jaccard – Cosine Similarity*
Nilai kemiripan dari kedua file dihitung dengan menggunakan 2 metode, yaitu:

- *Jaccard Similarity Coefficient*
  Metode ini diperkenalkan oleh Paul Jaccard, pada tahun 1901 [5, 6]. Metode ini menghitung kemiripan antara 2 himpunan di mana rumusnya seperti persamaan 1.

$$\textit{Jaccard's Similarity Coefficient (A,B)} = \frac{|A \cap B|}{|A \cap B|} \quad (1)$$

Dalam kasus *code plagiarism* rumusnya disesuaikan menjadi seperti pada persamaan 2 [2]

$$\textit{Jaccard's Similarity Coefficient (A,B)} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \quad (2)$$

Di mana:

f_{11} = jumlah blok N – gram yang ada di *source code* A dan *source code* B

f_{10} = jumlah blok N – gram yang hanya ada di *source code* A

f_{01} = jumlah blok N – gram yang hanya ada di *source code* B

- *Cosine Similarity Coefficient*

  Metode ini mengukur kemiripan antar 2 vektor dengan menghitung dot product antara dua buah N-gram yang dijadikan vector kolom [7]. Jika sudut yang terbentuk mendekati 0°, maka semakin mirip kedua blok N-gram yang dibandingkan.

  Jaccard's similarity coefficient memiliki keunggulan dalam plagiarism berbentuk banyak penambahan karakter atau kata–kata dalam *source code* hasil plagiaris, akan tetapi kelemahannya adalah *Jaccard similarity* membandingkan isi N – gram dengan eksak dan hanya melihat apakah ada suatu N – gram tertentu pada kedua file tersebut tanpa melihat posisi atau *sequence* penulisan yang berbeda sehingga *source code* yang berbeda secara penulisan namun sama secara *token* dapat terdeteksi sebagai hasil plagiarisme.

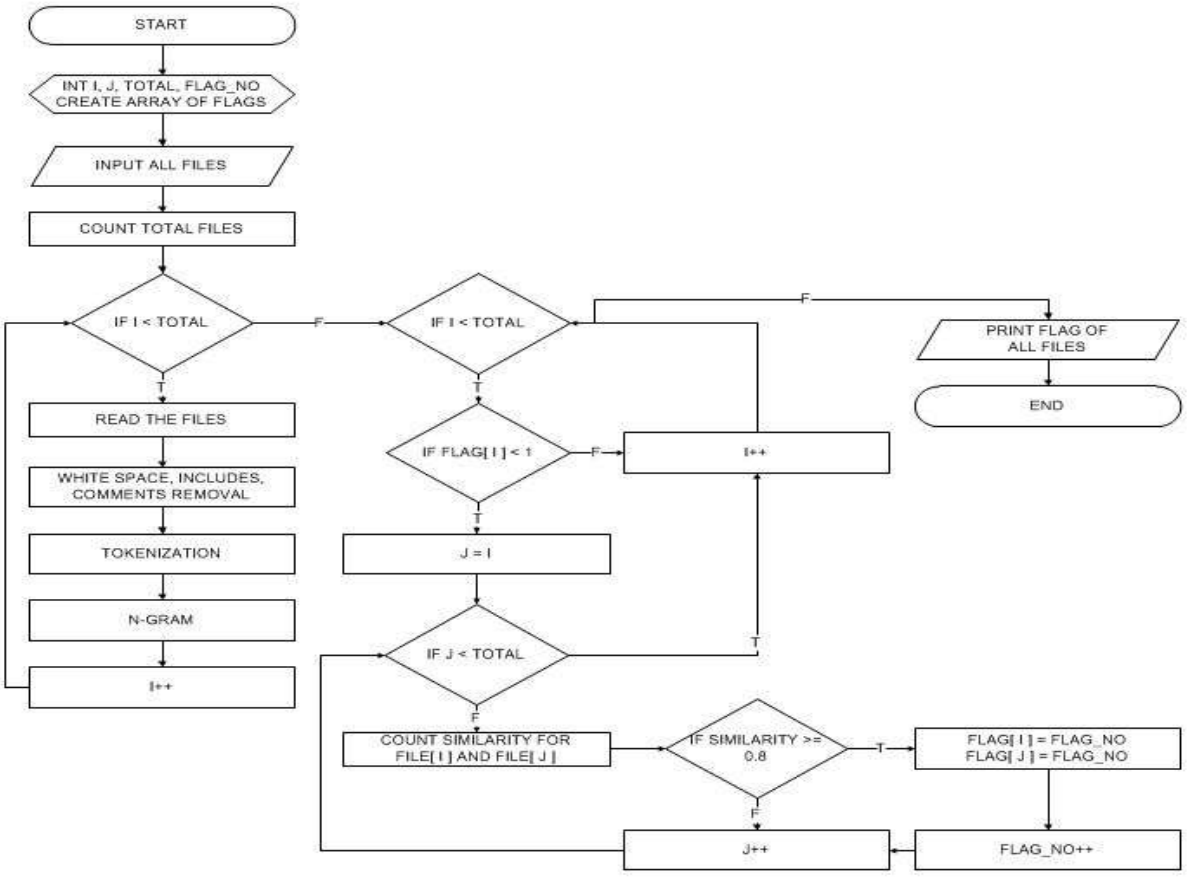
  Sedangkan *Cosine similarity coefficient* memiliki keunggulan dalam pengenalan plagiaris yang berupa penggeseran posisi minor karena cara kerja dari *Cosine similarity* adalah membandingkan N – gram yang sejajar satu sama lain pada kedua file. Akan tetapi apabila ada penggeseran mayor pada suatu *source code*, metode ini akan memunculkan hasil yang semakin menurun.

  Dalam aplikasi ini kedua metode tersebut digunakan untuk mendapatkan nilai kemiripan akhir. Rumus seperti pada persamaan 3.

$$\textit{Jaccard-Cosina Similarity} = \frac{\textit{Jaccard Similarity} + \textit{Cosina Similarity}}{2} \quad (3)$$

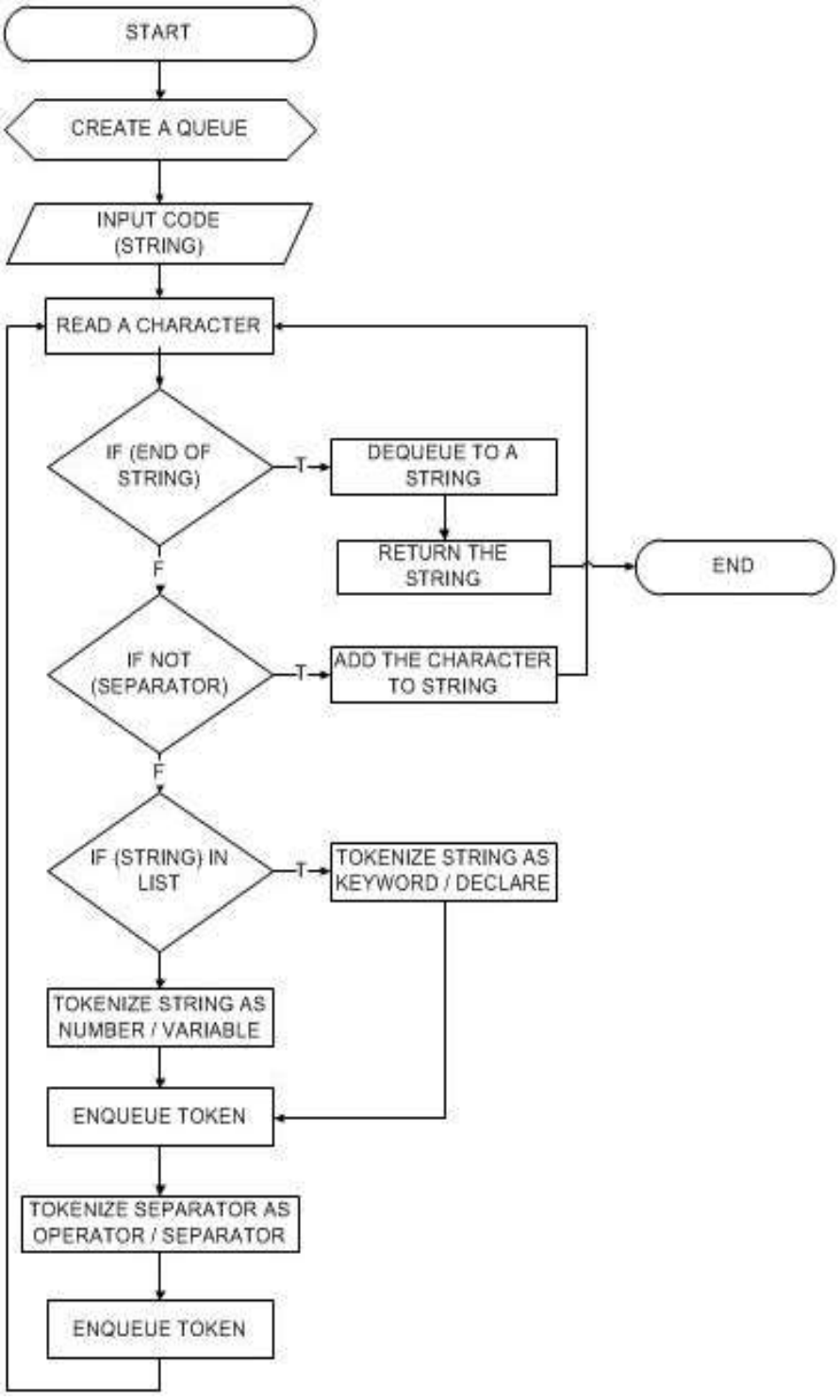
FLOWCHART

Algoritma Keseluruhan



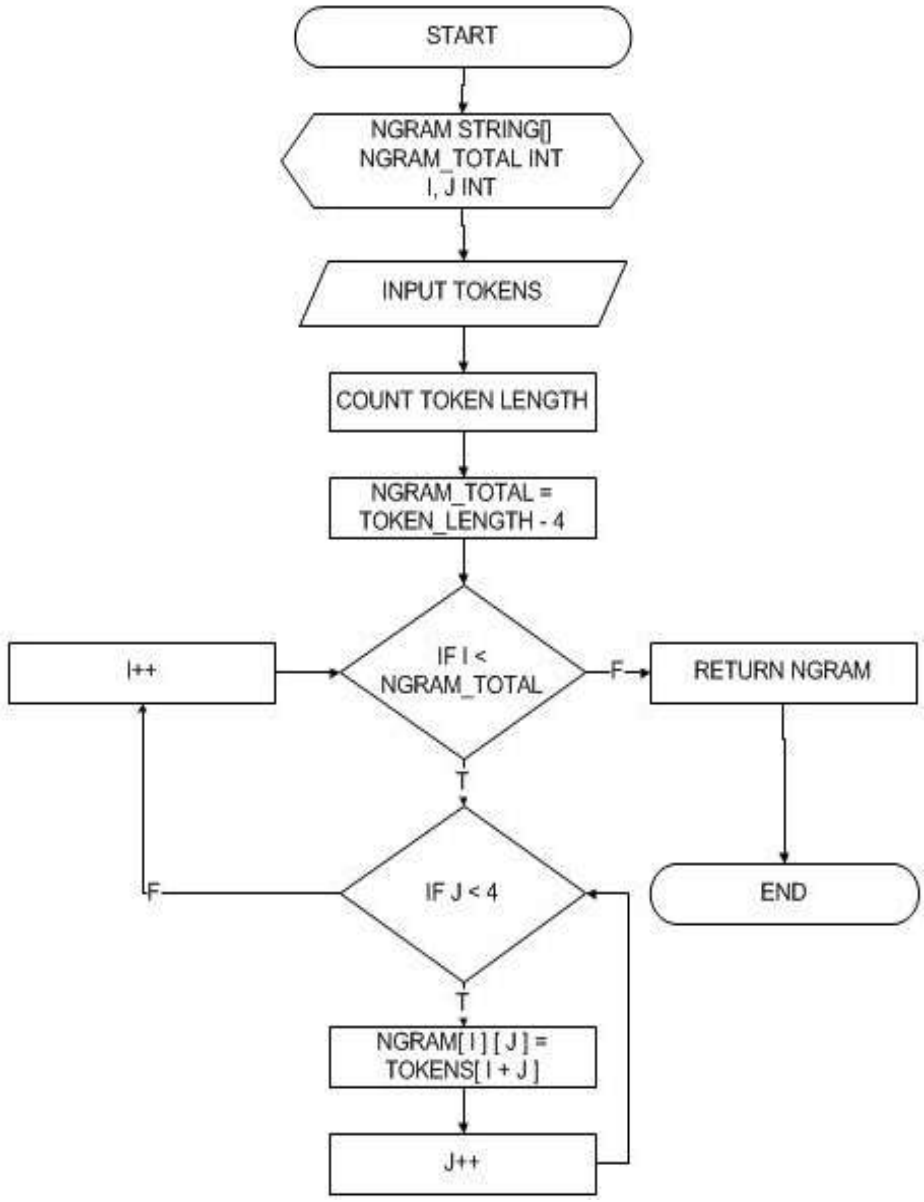
Gambar 2. Flowchart sistem secara keseluruhan

Algoritma *Tokenization*



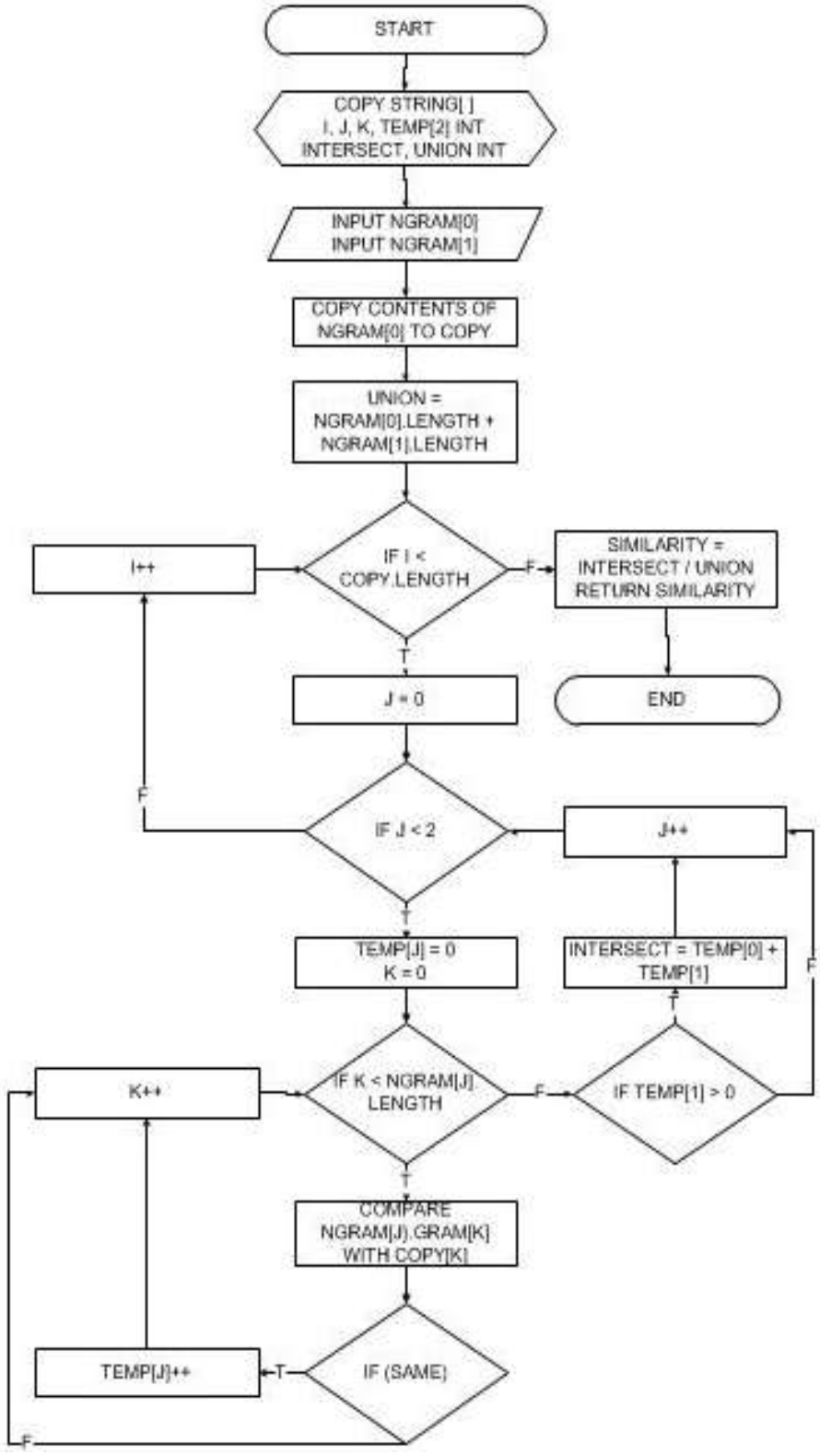
Gambar 3. Flowchart proses tokenization

Pembuatan *N – Gram*



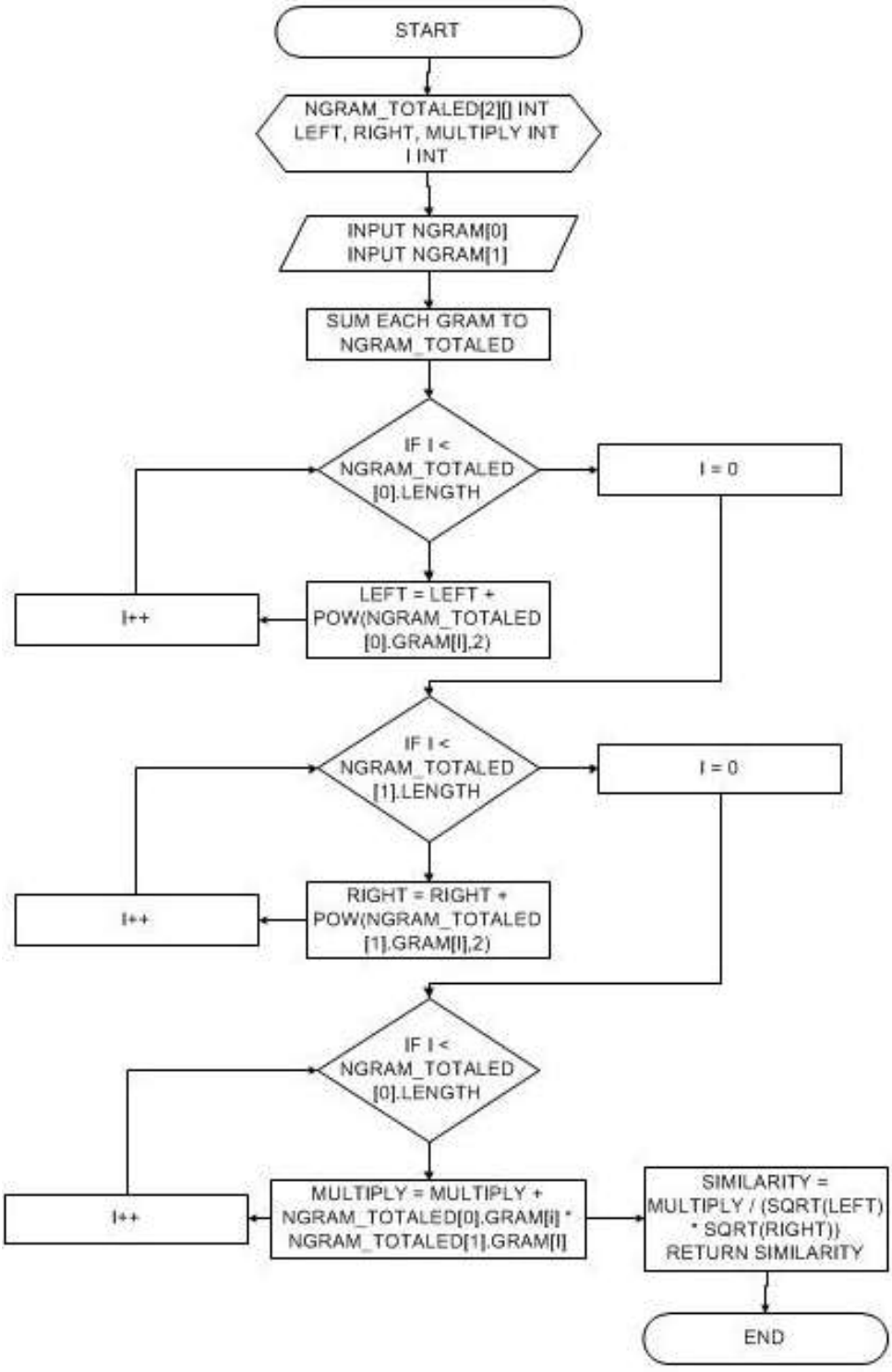
Gambar 4. Flowchart pembuatan N-Gram

Perhitungan *Jaccard Similarity Coefficient*



Gambar 5. Flowchart perhitungan koefisien dari Jaccard Similarity

Algoritma *Cosine Similarity Coefficient*



Gambar 6. Flowchart perhitungan koefisien dari cosine similarity

UJI COBA

Uji coba dilakukan dengan menggunakan 4 buah file source code yang bertujuan untuk menampilkan N buah bilangan prima pertama ke layar.

m26409013-main.cpp

```
int main()
{
    int i,j,k,times,n,number, counter;
    scanf("%d",&times);
    for(i=0;i<times;i++)
    {
        scanf("%d",&n);
        j=0;
        number=1;
        while(j<n)
        {
            number++;
            counter=0;
            for(k=1;k<=number;k++)
            {
                if(number%k==0)
                    counter++;
            }
            if(counter==2)
            {
                printf("%d",number);
                j++;
            }
        }
        printf("\n");
    }
    return 0;
}
```

m26409021-main.cpp

```
int main()
{
    int n,status,jumlah=0,data=2,testcase=0,now=0;
    scanf("%d",&testcase);
    while(now<testcase)
    {
        scanf("%d",&n);

        while(jumlah<n)
        {
            status=0;
            for(int j=2;j<data;j++)
            {
                if(data%j==0)
                    status=1;
            }
            if(status==0)
            {
                jumlah++;
                printf("%d",data);
            }
            data++;
        }
        printf("\n");
        now++;
        jumlah=0;
        data=2;
    }
    return 1;
}
```

m26409067-main.cpp

```
int main()
{
    int n,times;
    scanf("%d",&times);
    for(int j=0;j<times;j++)
    {
        scanf("%d",&n);
        int start=2,count=0;
        bool prime=true;
        while(count<n)
        {
            prime=true;
            for(int i=2;i<start;i++)
            {
                if(start%i==0)
                {
                    prime=false;
                    break;
                }
            }
            if(prime)
            {
                printf("%d",start);
                count++;
            }
            start++;
        }
    }
    return 0;
}
```

m26409085-main.cpp

```
int main()
{
    int banyak;
    scanf("%d", &banyak);

    int i=0, j=0, a, b;
    int n;
    while(i<banyak)
    {
        scanf("%d",&n);
        j=0;
        for(a=2; j<n; a++)
        {
            bool prima = true;
            for(b=2; b<a-1; b++)
            {
                if(a%b == 0) prima = false;
            }
            if(prima == true)
            {
                printf("%d",a);
                j++;
            }
        }
        printf("\n");
        i++;
    }
    return 0;
}
```

Gambar 7. Empat buah file source code yang mempunyai tujuan yang sama

Admin Page Check For Plagiarism

Submission History

NRP	PROBLEM	VERDICT
m26409013	Prime	Wrong Answer
m26409013	Prime	Wrong Answer
m26409013	Prime	Wrong Answer
m26409085	Prime	Compilation Error
m26409085	Prime	Compilation Error
m26409085	Prime	Compilation Error
m26409085	Prime	Wrong Answer
m26409013	Prime	Wrong Answer
m26409013	Prime	Correct
m26409085	Prime	Correct
m26409013	Prime	Wrong Answer
m26409013	Prime	Wrong Answer
m26409021	Prime	Compilation Error
m26409021	Prime	Wrong Answer
m26409021	Prime	Correct
m26409067	Prime	Wrong Answer
m26409067	Prime	Correct

Admin Page Check For Plagiarism

Submission History

FILE NAME	PROBLEM	FLAG
m26409013-main.cpp	Prime	0
m26409021-main.cpp	Prime	0
m26409067-main.cpp	Prime	0
m26409085-main.cpp	Prime	0

Gambar 8. Hasil penilaian jawaban yang benar (atas) deteksi plagiat pada empat buah file pada Gambar 7 (bawah).

Flag bernilai '0' menunjukkan terindikasinya kesamaan file tersebut dengan file lainnya.

KESIMPULAN

Aplikasi yang dibangun dapat melakukan penilaian secara otomatis terhadap file jawaban yang di-*upload* oleh peserta studio pemrograman dengan benar. Aplikasi ini juga mampu mendeteksi adanya plagiat dari dua file jawaban dari soal yang sama. Namun keakuratannya masih sebatas tipe plagiat yang dijelaskan pada subbab 2. Ada satu jenis lagi plagiat, yaitu kesamaan ide dan urutan perintah, namun ditulis dengan cara yang sedikit berbeda. Hal ini sering terjadi pada *loop* (perulangan). Ada dua jenis perintah *loop* yang bisa digunakan dalam bahasa C++, yaitu *for()* dan *while()*

Karena hasil kesamaan dinyatakan dalam bentuk presentase, maka jika presentase tinggi, kecurigaan adanya plagiat juga semakin tinggi. Ada kalanya sebuah kasus yang jarang terjadi, dua buah file yang bukan plagiat namun memiliki susunan *token* yang sama. Oleh karena itu, sampai saat ini, hasil dari aplikasi ini adalah sebuah daftar jawaban yang mempunyai kemungkinan adanya plagiat. Untuk memastikan apakah benar terjadinya plagiat, masih diperlukan koreksi manual. Aplikasi ini mengefisien-kan pekerjaan pengecekan ini, karena tidak perlu lagi mengecek keseluruhan file jawaban, namun hanya yang terindikasi saja.

DAFTAR PUSTAKA

1. GNU Compiler Collection, 2012, diakses dari http://en.wikipedia.org/wiki/GNU_Compiler_Collection Tanggal 10 April 2012.
2. g++, 2012, diakses dari http://www.cprogramming.com/g++.html Tanggal 5 Maret 2012.
3. Kusonpalalert, P., Nawairrigulchai, W. 2009. Source Code Plagiarism Detector. Bangkok: University of King Mongkut.
4. Haritha, N., Bhavani, M., Thammi Reddy, K. 2011. C Code Plagiarism Detection System. ISSN: 2221–8386.
5. Okiemute Omuta. ELECTRONIC SOURCE CODE PLAGIARISM DETECTION diakses dari http://eul.academia.edu/kheme/Papers/353119/Electronic_Source_Code_Plagiarism_Detection Tanggal 7 Maret 2012.
6. Jaccard Index, didownload dari http://en.wikipedia.org/wiki/Jaccard_index
7. Cosine Similarity, diakses dari http://en.wikipedia.org/wiki/Cosine_similarity Tanggal 2 Maret 2012.